Linear Nearest Neighbor Synthesis of Reversible Circuits by Graph Partitioning

Amlan Chakrabarti, Susmita Sur-Kolay, Senior Member, IEEE and Ayan Chaudhury

***Abstract*—Linear Nearest Neighbor (LNN) synthesis in reversible circuits has emerged as an important issue in terms of technological implementation for quantum computation. The objective is to obtain a LNN architecture with minimum gate cost. As achieving optimal synthesis is a hard problem, heuristic methods have been proposed in recent literature. In this work we present a graph partitioning based approach for LNN synthesis with reduction in circuit cost. In particular, the number of SWAP gates required to convert a given gate-level quantum circuit to its equivalent LNN configuration is minimized. Our algorithm determines the reordering of indices of the qubit line(s) for both single control and multiple controlled gates. Experimental results for placing the target qubits of Multiple Controlled Toffoli (MCT) library of benchmark circuits show a significant reduction in gate count and quantum gate cost compared to those of related research works.**

***Index Terms*—Linear Nearest Neighbor, Reversible circuit, Quantum Gates, Graph Partitioning, Quantum Cost.**

I. Introduction

QUANTUM computing is an emerging field of research in which the rules of quantum physics are used to solve certain computing problems more efficiently than any classical algorithm [1]. A quantum circuit is employed to process quantum bits (qubit), where a qubit is the unit of quantum information. It can be typically realized as a particular spin state of an electron, or a certain polarization state of a photon. While the two possible spin states of an electron are up ($\uparrow$) and $down$ ($\downarrow$), the two polarization states of a photon are $vertical$ ($\updownarrow$) and $horizontal$ ($\leftrightarrow$). Such states are denoted by $|0\rangle$ and $|1\rangle$ respectively. A qubit can also represent any superposition state $|\varphi\rangle$ = $\alpha|0\rangle$ + $\beta|1\rangle$ where $|\cdot\rangle$ is the standard Dirac notation [2] for quantum states, and α and β are the complex amplitudes representing the probabilities of state $|0\rangle$ and $|1\rangle$ respectively satisfying the condition $|\alpha|^2 + |\beta|^2 = 1$. Although a single qubit can theoretically be in infinite number of superposed states corresponding to all possible pair of values for the amplitudes α and β, in reality equal amplitudes for the two basis states lead to the observation that n qubits can represent at most 2^n possible superposed states. An *n*-qubit quantum state can be represented as $|\varphi\rangle = \sum_{i=0}^{2^n-1} \alpha_i|i\rangle$, where *i* is written in binary representation and $\sum_i |\alpha_i|^2 = 1$. This leads to the concept of a quantum register [3] with n qubits holding 2^n simultaneous values. Massive parallelism in quantum computing can occur if an operation is performed on the contents of a register, all possible values are operated on simultaneously. In practice however, the property of quantum de-coherence [4], [5] has to be tackled in order to achieve this sort of parallelism.

Quantum circuits or gate networks realizing Boolean functions are termed as Quantum Boolean Circuits (QBCs). Formally, a QBC is a quantum system of *N* qubits specified by $|x_1\rangle|x_2\rangle \ldots |x_N\rangle$, and a number of reversible quantum gates. The quantum gates such as NOT, Controlled-Not (CNOT), SWAP, which implement specific unitary operations manifest the logic operations.

Younnes and Miller [6] raised the issue that only the interaction (coupling) between physically adjacent qubits is desired for practical implementation of Quantum Boolean Circuits (QBCs). This is termed as the *Linear Nearest-Neighbor (LNN)* configuration. The requirement of nearest neighbor relationship between the control and the target qubits is truly justified due to the limitation of the J-coupling force [7] required to perform multi-qubit logic operations and this works effectively only between the adjacent qubits. In a QBC, pairs of SWAP gates play a key role in bringing the control and the target qubits of any quantum gate to adjacent lines. But this increases the gate cost. Hence, the aim in LNN synthesis of minimizing the number of additional SWAP gates has been addressed in this paper.

The layout of the paper is as follows. The next section is a review of existing synthesis approaches including methods for cost reduction. In Section III, the proposed LNN synthesis approach and related convention for multi-controlled gate decompositions are discussed. In Section IV, the graph partitioning based LNN synthesis method is presented. Experimental results appear next and concluding remarks in Section VI.

II. Related Works

One of the major challenges in hardware implementation by any technology is to minimize power dissipation. Landauer [8] showed that irreversible circuits must dissipate certain energy irrespective of the technology used. Bennett [9] in his work clearly established that the most effective way of implementing reversible circuits to avoid this situation is to employ quantum technology. Furthermore, using quantum technology certain computational tasks can be performed exponentially faster [10] than by the hardware of existing irreversible technologies.

In the implementation of quantum circuits, optimization of circuit levels as well gate count in a quantum boolean network

Amlan Chakrabarti is with A.K.Choudhury School of I.T., University of Calcutta, 92, A.P.C.Road, Calcutta 700 009, India. e-mail: acakcs@caluniv.ac.in

Susmita Sur-Kolay is with Advanced Computing & Microelectronics Unit, Indian Statistical Institute, 203, B.T.Road, Calcutta 700 108, India. e-mail: ssk@isical.ac.in

Ayan Chaudhury is with the Department of Computer Science & Engineering, University of Calcutta, 92, A.P.C. Road, Calcutta 700 009 , India. e-mail: ayanchaudhury.cs@gmail.com

needs to be done. For a small circuit, this can be done easily, but for the larger ones, we need to find a proper algorithmic approach for minimizing quantum cost in the circuit. Here, quantum cost means the cost in implementing a given quantum circuit using a suitable technology.

Synthesis of quantum boolean network is an emerging research area and some fine works [11], [12], [13], [14] have been done on reversible circuit synthesis in recent years. Local optimization methods have been proposed recently in [15], [16]. Other models on reversible logic synthesis have also been proposed by many authors [17], [18], [19], [20]. But from a practical viewpoint there are other drawbacks with these approaches. Each gate must be realized in a Linear Nearest Neighbor architecture. Wille *et al.* [21] have discussed in detail about the cost metrics and the need for LNN optimization. Hirata *et al.* [22] discusses the schemes for permuting the qubits for achieving the LNN configuration. It carries out the optimization in the number of SWAP gates by carrying out the search for each gate individually, a sort of local ordering. It is not shown how this can lead to the optimization in the number of SWAP gates considering the whole circuit. Related works can also be found in [23]. Most of these methods are built for Linear Nearest Neighbor (LNN) architecture where only adjacent qubits can interact. In most of the articles, the authors aim at minimizing nearest neighbor cost to achieve NNC optimality. Wille *et al.* [21] showed that achieving NNC optimality by introducing additional SWAP gates increases quantum cost. They aim optimality by decomposing a TOFFOLI gate to basic gates, e.g, NOT, CNOT, Controlled-V and Controlled-V^+ and then reordered the circuit lines by means of local and global reordering. This results in NNC optimality and reduced quantum cost. Maslov [24] presents a theoretical study on quantum circuit placement in physical hardware by means of a graph theoretic approach. A TOFFOLI network synthesis using template matching has also been proposed in [14].

In this paper we propose a new approach for inserting pairs of SWAP gates, and reordering the qubit lines based on a graph partitioning approach. This gives better results compared to those by the earlier ones including [21].

III. Linear Nearest Neighbor Synthesis

Our work is based on reordering the qubit lines so that the interacting qubits are adjacent to each other, i.e., the distance between target and control lines are minimized (this is termed as Nearest Neighbor Cost (NNC) in [21]). Without loss of generality, it may be assumed that a given QBC is not in nearest neighbor form. SWAP gates are inserted appropirately to convert the QBC to a corresponding LNN architecture. It may be noted that for 1-qubit and 2-qubit gates, LNN architecture can be obtained without any additional SWAP gates. Hence, the multi-qubit gates need special attention in LNN synthesis to reduce the number of extra SWAP gates required,

In Fig. 1(a), a Toffoli or C^kNOT quantum gate is shown which is not in the nearest neighbor architecture. Insertion of SWAP gates for LNN without reordering the qubit lines is shown to be non-optimal in Fig. 1(b), whereas the optimal synthesis after reordering as in Fig. 1(c) requires no SWAP gates.

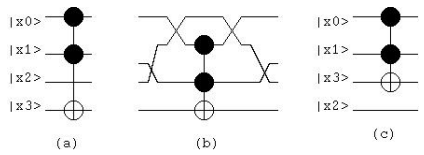

Fig. 1. LNN synthesis of TOFFOLI (a) a single TOFFOLI gate, (b) introduction of SWAP gates resulting 4 SWAPs. (c) is the reordered circuit resulting zero SWAP gate requirement

In this example of Fig. 1, we have considered a single gate where the reordering can be done easily. The pertinent question is how to solve this for large benchmark QBC circuits with many qubit lines. Furthermore, an optimal solution with minimum NNC is the desideratum.

For ease of handling, we further sub-divide the LNN synthesis problem for QBCs as one for NOT, CNOT and TOFFOLI (NCT) and another for multiple controlled TOFFOLI (MCT) gates. In the next two subsections we discuss these two types of synthesis separately.

A. LNN Synthesis for NCT gate library

For the realization of quantum circuits in hardware technology we have to introduce SWAP gates or a chain of basic quantum gates in the circuit to make it a LNN circuit. For introducing SWAP gates in a TOFFOLI network to make a LNN architecture, an algorithm for counting the required SWAP gate pairs is introduced here.

For a simple NOT gate, the number of SWAP pair required is zero. For a CNOT gate, the number of SWAP pairs is simply the number of intermediate qubit lines between top and bottom control lines. There may be two variations of a CNOT gate as shown in Fig. 2(a) and 2(b).

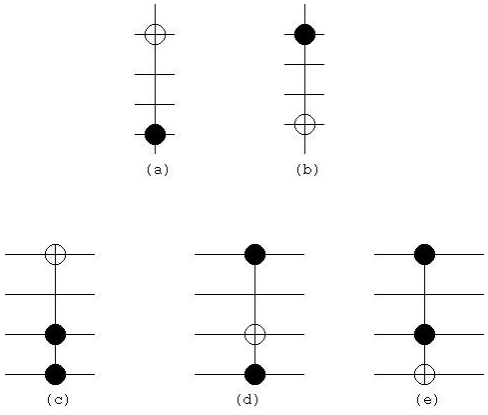

Fig. 2. Variations of CNOT and TOFFOLI. (a), (b) are two types of CNOT. (c), (d), (e) are variations of TOFFOLI

For a TOFFOLI gate the scenario is somewhat complex. There may be three variations of a TOFFOLI gate corresponding to the positions of its target qubit. All the variations of CNOT and TOFFOLI are shown in Fig. 2(c), (d), (e).

1) Rules for counting SWAP pairs: We now formally present the rule for counting the additional SWAP gates needed to convert a non-LNN gate to the corresponding LNN. Let $ctrl_1$, $ctrl_2$, $target$ denote the two control qubit lines and the target output qubit line of a TOFFOLI gate. The input qubit lines have consecutive integer indices starting with 0 for the topmost qubit line. The control qubit line having lower index value is denoted by $ctrl_1$ and the other control qubit line by $ctrl_2$.

Rules for counting SWAP pairs

Case 1: $target < ctrl_1 < ctrl_2$
if $ctrl_1 - target > 1$, then required SWAP pair is $s_1 + s_2$ where $s_1 = ctrl_1 - target - 1$ and $s_2 = ctrl_2 - target - 2$
else if $ctrl_2 - ctrl_1 > 1$, then required SWAP pair is s where $s = ctrl_2 - ctrl_1 - 1$

Case 2: $ctrl_1 < target < ctrl_2$
if $target - ctrl_1 > 1$, then required SWAP pair is s where $s = target - ctrl_1 - 1$, otherwise no SWAP pair is needed
if $ctrl_2 - target > 1$, then required SWAP pair is s where $s = ctrl_2 - target - 1$, otherwise no SWAP pair is needed

Case 3: $ctrl_1 < ctrl_2 < target$
if $target - ctrl_2 > 1$, then required SWAP pair is $s_1 + s_2$ where $s_1 = target - ctrl2 - 1$ and $s_2 = target - ctrl_1 - 2$
else if $ctrl_2 - ctrl_1 > 1$, then required SWAP pair is s where $s = ctrl_2 - ctrl_1 - 1$

B. LNN Synthesis for MCT gate library

Another side of the problem is to decompose each MCT gate in the reversible circuit into an equivalent NCT network. After then the nearest neighbor synthesis is to be done for the entire circuit. Consider the MCT gate in Fig. 3.

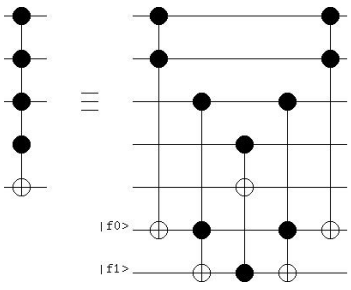

Fig. 3. Decomposition of MCT gate: Replacement of a C^4NOT gate by equivalent TOFFOLI with two ancillary qubits f_0,f_1

Here, the original circuit is a multiple controlled TOFFOLI. Using standard decomposition [23], [25] the gate is converted to equivalent TOFFOLI. As a result, gate count and quantum cost has increased. Two extra ancillary qubits have come into the scenario and the circuit is not in nearest neighbor condition.

As far as we are focussing the hardware implementation of quantum circuits, the decomposition of an arbitrary circuit (Multiple controlled TOFFOLI) is necessary. Perkowski *et al.* [23] states the necessity of decomposition and introduction of ancillary qubits. Wille *et al.* [21] showed some decomposition leading to increased quantum cost by adding SWAP gates whenever a non-adjacent quantum gate appears.

In our approach we have taken circuit files from *Revlib* [26] and written a C program to parse it into an equivalent TOFFOLI network. It can be shown that for decomposition of a single C^kNOT gate, the number of TOFFOLI required is $2(k-2)+1$ and number of ancillary qubits required is $k-2$ [25]. Using this formula we have simply reconstructed the decomposed circuit. The resultant circuit contains only NOT, CNOT and TOFFOLI gates.

IV. Reordering by Graph Partitioning

In this section, we propose a graph partitioning based approach to get the ordering of qubit lines to achieve LNN architecture. Any non-LNN circuit can be converted to LNN one by introducing additional SWAP gates. But the linear ordering of the qubit lines will make the difference in number of SWAP gates required in making the LNN circuit. On a practical quantum computer, a number of SWAP operations is necessary to emulate the behavior of a quantum circuit running on an ideal machine [27]. Some works on TOFFOLI network synthesis can be found in [28], [29]. Some authors focused on logic synthesis by permutation of qubit lines [30], [31] also. Very recently LNN synthesis by means of line reordering has presented by Saeedi *et al.* [32].

However, in this section we propose a balanced graph partitioning approach through which reordering can be achieved with less overhead producing better result than the existing ones. Although graph partitioning problem is NP-complete, many algorithms exist that result a reasonably good partition. We first formulate the graph partitioning problem mathematically and then introduce the method how we have exploited the problem in NN synthesis.

1) Qubit line adjacency graph: We define an undirected weighted *qubit line adjacency graph* $G = (V, E, w : E \rightarrow \mathcal{N})$ for a given QBC C where each vertex v represents an input qubit line. There is an edge between two vertices u and v if the corresponding qubit lines appear as the control and target qubit lines of a single gate in C. Thus, for a NOT gate there is no edge; for a CNOT gate there is an edge between its control and target qubit lines; for a TOFFOLI$(ctrl_1, ctrl_2, target)$ gate there are two edges namely ($ctrl_1$,$target$) and ($ctrl_2, target$). The weight $w(e)$ of an edge $e = (u, v)$ between the vertices u and v is the number of times the pair of qubit lines corresponding to u and v appear as the control and target qubit of any quantum gate in the given QBC. It may noted that for a QBC with a single output, this graph is connected. Fig. 4 illustrates a QBC of NCT gate library and its qubit line adjacency graph.

The problem of re-ordering the input qubit lines in a QBC C is reduced to finding a linear ordering $f : V \rightarrow \{1, 2, \ldots, |V|\}$ of the vertices of its qubit line adjacency graph G such that $\Sigma_{e=(u,v)\in E}|f(u)-f(v)|$ is minimum. This is known to be NP-complete [33]. There are approximate algorithms for solving

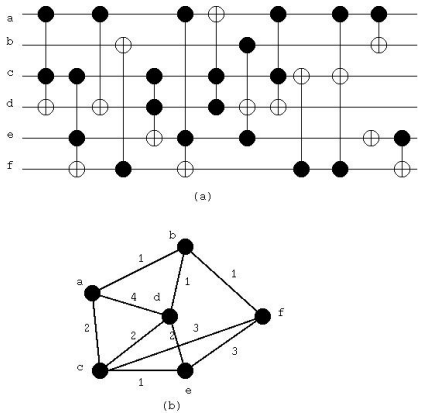

Fig. 4. Graph formation: (a) is the TOFFOLI network having NOT, CNOT & TOFFOLI gates, (b) is the weighted graph formed from this circuit

these [34] which are non-trivial to implement. Fortunately, it was also shown that the heuristic for getting the linear ordering by listing the leaves of the binary tree representing recursive balanced bi-partitioning of the graph, is good [35]. We therefore, adopt this line of approach to solve the problem of re-ordering the qubit lines of a given QBC so that the NNC is minimal.

Various algorithms exist for solving graph partitioning problem in polynomial time, having their relative advantages and disadvantages. Many of these methods compute the eigenvector corresponding to the second smallest eigenvalue [36], [37]. But these methods are expensive with respect to running time. Geometric partitioning methods [38], [39] need greater execution time. Moreover, quality of partitions are also not up to the mark due to randomized nature of partitioning of these approaches.

2) Reordering by partitioning: We have used *pmetis* [40] graph partitioning tool for partitioning. This tool is developed using multilevel partitioning algorithm (Coarsening, Partitioning and uncoarsening phases) and produce good quality of partition with less execution time. The complexity of the algorithm used in *pmetis* is approximately $O((n+m)*log(k))$ where n is the number of nodes, m is the number of edges and k is the number of partitions.

We have written a C program to parse the circuit description of a TOFFOLI network to the graph description format required for *pmetis* input. The output is a linear ordering of the partitions that are returned by *pmetis*. But in most of the cases the ordering returned by *pmetis* there are fewer non-empty partitions than the required number of partitions. Hence, more than one vertices may present in a single partition. To resolve this problem, we have made a linear ordering of the partition order. For example, if the input qubit lines are 0 through 5 and the partitioning order is: $2, 1, 1, 3, 2, 4$, we will reorder these as: $2, 0, 1, 4, 3, 5$.

From the reordered qubit lines we have again reconstructed the NCT network. Then the required number of SWAP pair is calculated in both cases using the rule presented in the previous section. Figure 5 demonstrates the synthesis of the circuit *4mod5-bdd_287* from *RevLib*. The original circuit is represented in Fig.5(a), which is not in NN architecture. The

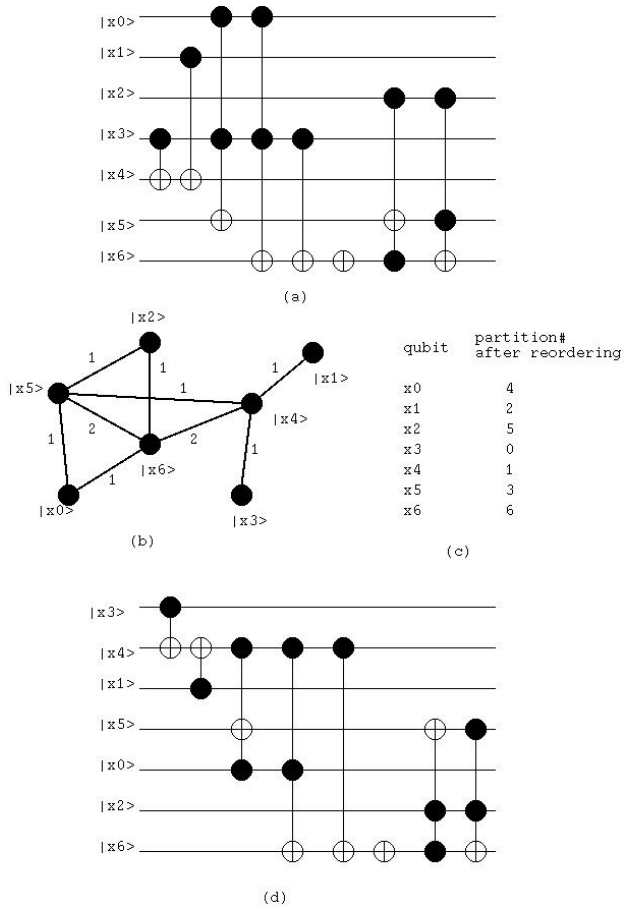

Fig. 5. Synthesis of the circuit *4mod5-bdd_287*. (a) the original circuit which needs 15 SWAP pairs for LNN architecture, (b) the graph formed from this circuit, (c) the ordering of qubit lines after partitioning, (d) is the reconstructed circuit from new ordering, with only 12 SWAP pairs for LNN architecture

number of SWAP gates required to make this circuit LNN is 15. Fig.5(b) demonstrates the formation of the graph from the circuit and 5(c) is the partition numbers returned by *pmetis* after the partitioning. Fig.5(d) is the reconstructed circuit after the reordering, which clearly shows that number of SWAP gate requirement has come down to 12 to make it an LNN circuit.

V. Experimental Results

The experimental results are shown in Table I. The first column is the list of benchmark circuits used in our experiment, next three columns are the number of qubit lines (N), gate count (GC) and quantum cost (QC) in the original circuit. SWAP cost and total quantum cost (QC) of the decomposed circuit before ordering are shown in the next two columns, followed by SWAP cost and QC after ordering. Then the percentage reduction of quantum cost after reordering the qubit lines are shown. Next column denotes the results reported by Wille *et al.* [21]. The percentage decrease in quantum cost over [21] is shown in the last column.

Regarding the quantum cost of various gates, cost of NOT, CNOT and SWAP gates are taken as $1, 5$ and 3 respectively [21]. When two qubits of a particular gate are adjacent, then there is no need to introduce SWAP between them, when two qubit lines are adjacent then only the notion of introducing

SWAP gates comes into the scenario. We have taken 54 benchmark circuits from *RevLib*, out of which the results for 20 circuits are also reported in [21]. It can be seen that using our method we can reduce the number of SWAP gates and therefore the total quantum cost by 17.5% in average compared to [21], except for a few of the cases in which ordering of the qubit lines have worsened the situation. The execution times of the algorithm are also negligible. The percentage decrease in quantum cost due to the reordering of qubit lines, considering all the circuits as given in Table I is about 47%.

VI. Conclusion

In this work, we have focussed on linear nearest neighbor architecture where SWAP gates are introduced between the qubit lines whenever it is not in NN architecture. The method proposed reduces the SWAP gate count significantly for the RevLib benchmark circuits. It has also led to an appreciable decrease in the overall quantum cost for most of the benchmark circuits compared to the related works, and by 47% on an average. This method for converting a QBC with CNOT, C^2NOT and even MCT gates into their nearest neighbor equivalent form can be utilized for the development of low-level circuit synthesis automation tools in the quantum computing domain.

References

[1] P. W. Shor, "*Quantum Computing*", Documenta Mathematica-Extra Volume, ICM, pp. 1-1000, 1998.

[2] R. P. Feynman, "*Quantum Mechanical Computers,*", Foundations of Physics, Vol.16, pp.507-531, 1986.

[3] M. A. Nielsen and Isaac L. Chuang, "*Quantum Computation and Quantum Information*", Cambridge University Press, 2002.

[4] J. Preskil, "*Logical reversibility of computation,*", I.B.M. J. Res. Dev., Vol.17, pp. 525-532, 1973.

[5] H. K. Lo, S. Popescu and T. Spiller, "*Introduction to quantum computation and information,*", Singapore : World Scientific Publ., 1999.

[6] A. Younes and J. Miller, "*Representation of Boolean Quantum Circuits as Reed Muller Expressions,*", International Journal of Electronics, Volume 91, No. 7, pp. 431-444, July 2004.

[7] A. Grigorenko and D. V. Khveshchenko, "*Single Step Implementation of Universal Quantum Gates,*", Physical Review Letters, 95.110501, 2005.

[8] R. Landauer, "*Irreversibility and heat generation in the computing process,*", IBM Journal of Research and Development, 5:183-191, July 1961.

[9] C. Bennett., "*Logical reversibility of computation,*", I.B.M. J. Res. Dev., Vol.17, pp. 525-532, 1973.

[10] Shor, P. (1994) "*Algorithms for quantum computation: Discrete logarithms and factoring,*", Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, pp. 124-134

[11] D. M. Miller, "*Spectral and two-place decomposition techniques in reversible logic,*", In Midwest Symposium on Circuits and Systems, August 2002.

[12] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "*Synthesis of reversible logic circuits,*", IEEE Transactions on CAD, 22(6):723-729, June 2003.

[13] D. Maslov and G. W. Dueck, "*Reversible cascades with minimal garbage,*", IEEE Transactions on CAD, 23(11):1497-1509, November 2004.

[14] D. Maslov, G. W. Dueck, and D. M. Miller, "*Toffoli network synthesis with templates,*", IEEE Transactions on CAD, 24(6):807-817, June 2005.

[15] K. Iwama, Y. Kambayashi, and S. Yamashita, "*Transformation rules for designing CNOT-based quantum circuits,*", In Design Automation Conference, pages 419-424, New Orleans, Louisiana, USA, June 2002.

[16] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "*Graph based simulation of quantum computation in the state vector and density matrix representation,*", Quantum Information and Computation, Vol. 5, No. 2, pp. 113-130, 2005.

[17] D. M. Miller, D. Maslov, and G. W. Dueck, "*Synthesis of quantum multiple valued circuits,*", Journal of Multiple-Valued Logic and Soft Computing, vol. 12, no. 5-6, pp. 431-450, 2006.

[18] G. Yang, W. Hung, X. Song, and M. Perkowski, "*Majority-based reversible logic gates,*", Theoretical Computer Science, vol. 334, no. 1-3, 2005.

[19] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "*Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis,*", IEEE Transactions on CAD, vol. 25, no. 9, pp. 1652-1663, 2006.

[20] W. N. Alhagi, M. Hawash, and M. Perkowski, "*Synthesis of Reversible Circuits for Large Reversible Functions,*", Facta Universitais, Series: Electronics and Energetics, vol. 24, no. 3, pp. 273-289, December 2010.

[21] R. Wille, M. Saeedi and R. Drechsler, "*Synthesis of Reversible Functions Beyond Gate Count and Quantum Cost,*", International Workshop on Logic Synthesis (IWLS), USA, 2009.

[22] Y. Hirata, M. Nakanishi, and S. Yamashita, "*An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture,*", 3rd International conference on Quantum, Nano, and Micro Technologies, 2009.

[23] M. Perkowski, M. Lukac, D. Shah, and M. Kameyama, "*Synthesis of quantum circuits in Linear Nearest Neighbor model using Positive Davio Lattices,*", Facta Universitais, Series: Electronics and Energetics, vol. 24, no. 1, pp. 71-87, April 2011.

[24] D. Maslov, S.M. Falconer, and M. Mosca, "*Quantum Circuit Placement,*", IEEE Transactions on CAD, Volume. 27, No. 4, 752 - 763, April 2008.

[25] A. Chakrabarti and S. Sur-Kolay, "*Nearest Neighbour based Synthesis of Quantum Boolean Circuits,*", Engineering Letters, vol. 15, no. 2, 2007.

[26] http://www.revlib.org/

[27] B.S. Choi and R. Van Meter, "*Effects of Interaction Distance on Quantum Addition Circuits,*", ACM Journal on Emerging Technologies in Computing Systems, Volume 7, Issue 3, August 2011.

[28] D. Groe, X. Chen, G.W. Dueck, R. Drechsler, "*Exact SAT-based Toffoli Network Synthesis,*", Great Lakes Symposium on VLSI (GLSVLSI), pp. 96-101, Stresa, 2007.

[29] D. Maslov, D. M. Miller, G. W. Dueck, "*Techniques for the Synthesis of Reversible Toffoli Networks,*", ACM Transactions on Design Automation of Electronic Systems, Volume 12, Issue 4, September 2007 .

[30] R. Wille, D. Groe, G.W. Dueck, R. Drechsler, "*Reversible Logic Synthesis with Output Permutation,*", 22nd International Conference on VLSI Design, pp. 189-194, New Delhi, 2009.

[31] D.M. Miller, D. Maslov, G. W. Dueck, "*A Transformation Based Algorithm for Reversible Logic Synthesis,*", Proceedings of the 40th annual Design Automation Conference(DAC), June 2-6, 2003, Anaheim, California, USA.

[32] M. Saeedi, R. Wille, and R. Drechsler, "*Synthesis of quantum circuits for linear nearest neighbor architectures,*", Quantum Information Processing, Volume 10 Issue 3, June 2011.

[33] M. R. Garey And D. S. Johnson 1979, "*Computers and Intractability: A Guide to Theory of NP Completeness,*", W. H. Freeman and Co., San Francisco, CA.

[34] U. Feige and J. R. Lee, "*An improved approximation ratio for the minimum linear arrangement problem,*", Inform. Process. Lett. 101, 1, 26-29, 2007.

[35] G. Even, Joseph S. Naor, S. Rao and B. Schieber, "*Divide-and-conquer approximation algorithms via spreading metrics,*" Journal of the ACM (JACM), vol. 47, no. 4, pp. 585-616, July 2000.

[36] A. Pothen, H. D. Simon, L. Wang, and S. T. Bernard, "*Towards a fast implementation of spectral nested dissection,*", In Supercomputing '92 Proceedings, pages 42-51, 1992.

[37] A. Pothen, H. D. Simon, and K. P. Liou, "*Partitioning sparse matrices with eigenvectors of graphs,*", SIAM Journal of Matrix Analysis and Applications, 11(3):430-452, 1990.

[38] M. T. Heath and P. Raghavan, "*A Cartesian parallel nested dissection algorithm,*", SIAM Journal of Matrix Analysis and Applications, 16(1):235-253, 1995.

[39] G. L. Miller, S. H. Teng, and S. A. Vavasis, "*A unified geometric approach to graph separators,*", In Proceedings of 31st Annual Symposium on Foundations of Computer Science, pages 538-547, 1991.

[40] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview/.

TABLE I
COMPARISON OF QUANTUM COST FOR PROPOSED LNN SYNTHESIS METHOD VS. [21] ON REVLIB BENCHMARKS

BENCHMARK CIRCUIT	ORIGINAL CIRCUIT			BEFORE ORDERING		AFTER ORDERING		% DECREASE IN QC DUE TO REORDERING	QC REPORTED BY [21]	% DECREASE IN QC OVER [21]
	N	GC	QC	SWAP COST	TOTAL QC	SWAP COST	TOTAL QC			
3_17_13	3	6	14	6	20	0	**14**	30	28	50
4_49_17	4	12	16	66	98	18	**50**	48.9	98	48.9
4gt4-v0_80	5	5	37	120	153	48	**81**	47	138	41.3
4gt5_75	5	5	21	78	101	36	**59**	41.5	79	25.3
4gt12-v1_89	5	5	45	114	**157**	126	169	-7.64	168	6.5
4gt13-v1_93	5	4	16	72	90	6	**24**	73.3	53	54.7
4gt-10v1_81	5	6	34	120	158	96	**134**	15.2	147	8.8
4mod5-bdd_287	7	8	24	90	114	72	**96**	15.7	-	-
4mod5-v1_23	5	8	24	84	108	42	**66**	38.8	78	15.3
5xp1_194	17	85	1430	28194	29523	7164	**8493**	71.2	-	-
9symml_195	10	129	14193	34458	38303	19764	**23609**	38.3	-	-
add6_196	19	229	6455	122910	128831	45606	**51527**	60	-	-
adr4_197	13	55	727	9774	10489	4986	**5701**	45.6	-	-
aj-e11_165	4	13	45	84	**131**	90	137	-4.5	181	27.6
alu1_198	20	32	228	6498	6756	1944	**2202**	67.4	-	-
alu2_199	16	157	5654	70716	74991	21240	**25515**	65.9	-	-
alu3_200	18	94	2632	45954	48290	21000	**23336**	51.6	-	-
alu4_201	22	1063	55388	1059834	1106423	541380	**587969**	46.8	-	-
alu-bdd_288	7	9	29	144	173	102	**131**	24.2	-	-
apla_203	22	80	3438	77742	80828	23394	**26480**	67.2	-	-
apex4_202	28	5376	237963	7659894	7875016	1922358	**2137480**	72.8	-	-
bw_291	87	307	943	79326	80269	46158	**47101**	41.3	-	-
c7552_205	21	80	1728	50418	52102	10746	**12430**	76.1	-	-
clip_206	14	174	6731	72792	77764	29292	**34264**	55.9	-	-
cm42a_207	14	35	377	7236	7617	1590	**1971**	74.7	-	-
cm85a_209	14	69	2252	26958	28995	11562	**13599**	54	-	-
cm150a_210	22	53	1096	8472	9467	5124	**6119**	35.3	-	-
cm151a_211	28	33	888	21216	22027	5016	**5827**	73.5	-	-
cm152a_212	12	16	252	1566	1816	1416	**1666**	8.2	-	-
cm163a_213	29	39	756	22800	23499	6780	**7479**	68.1	-	-
cmb_214	20	18	910	7800	8234	2748	**3182**	61.3	-	-
co14_215	15	30	3488	24438	26064	8466	**10092**	61.2	-	-
cu_219	25	40	1148	30234	31262	9402	**10430**	66.6	-	-
cycle10_2_110	12	19	1202	9690	10417	2808	**3535**	66	8046	56
dc1_220	11	39	416	5994	6419	1482	**1907**	70.2	-	-
dc2_222	15	75	1886	32052	33809	8634	**10391**	69.2	-	-
decod24-v3_46	4	9	9	54	63	12	**21**	66.6	**21**	0
decod_217	21	80	1728	50418	52102	10746	**12430**	76.1	-	-
dist_223	13	185	7601	76164	81591	24828	**30255**	62.9	-	-
f51m_233	22	663	37400	612030	639493	281508	**308971**	51.6	-	-
ham15_108	15	70	453	3312	3764	2418	2870	23.7	**2588**	-10.8
hwb4_52	4	11	23	42	**65**	48	71	-9.2	**65**	0
hwb5_55	5	24	104	378	492	276	390	20.7	**337**	-15.7
inc_237	16	93	2140	40410	42407	11772	**13769**	67.5	-	-
mod5adder_128	6	15	83	522	613	150	**241**	60.6	330	26.9
mod8-10_177	5	14	94	330	418	234	**322**	22.9	363	11.2
plus127mod8192_162	13	910	73357	508134	551588	403938	**447392**	18.8	503516	11.1
plus63mod4096_163	12	429	32539	193446	211559	159258	**177371**	16.1	210400	15.6
plus63mod8192_164	13	492	45025	267480	290798	203772	**227090**	21.9	279016	18.6
rd53_135	7	16	77	558	636	348	426	33	**303**	-40.5
rd84_313	34	104	304	6780	7084	4548	**4852**	31.5	-	-
sqn_258	10	76	2122	15258	16784	5562	**7088**	57.7	-	-
sym9_317	27	62	206	3870	4076	1686	**1892**	53.5	-	-
z4ml_269	11	48	642	7386	8018	3600	**4232**	47.2	-	-
AVERAGE COST REDUCTION %								**46.6**		**17.5**